PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04B

A2

(11) International Publication Number: WO 99/17458

(43) International Publication Date: 8 April 1999 (08.04.99)

(21) International Application Number: PCT/US98/20414

(22) International Filing Date: 30 September 1998 (30.09.98)

(30) Priority Data:

60/060,418	30 September 1997 (30.09.97)	US
09/149,751	8 September 1998 (08.09.98)	US

(71) Applicant: TANDEM COMPUTERS INCORPORATED [US/US]; 10435 North Tantau Avenue, Cupertino, CA 95014 (US).

(72) Inventors: JOHNSON, Charles, S.; 3954 Williams Road, San Jose, CA 95117 (US). SHAFIQ, Muhammad; P.O. Box 1193, El Granada, CA 94018 (US).

(74) Agents: BENNETT, Robert, J. et al.; Townsend and Townsend and Crew, 8th floor, Two Embarcadero Center, San Francisco, CA 94111-3834 (US).

(81) Designated States: JP, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published

Without international search report and to be republished upon receipt of that report.

(54) Title: TRANSACTION STATE BROADCAST METHOD USING A TWO-STAGE MULTICAST IN A MULTIPLE PROCESSOR CLUSTER

(57) Abstract

In a multiple processing system comprising multiple communicatively interconnected nodes, each node having one or more processor units, multicast messages sent by a sender node will contain information that allows intended receiver nodes to check and determine the possibility that earlier-sent multicast messages from the sender node were not received by the receiver node.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

TRANSACTION STATE BROADCAST METHOD USING A TWO-STAGE MULTICAST IN A MULTIPLE PROCESSOR CLUSTER

BACKGROUND OF THE INVENTION

The present invention relates generally to a computing system using clustering principles, and more particularly to transmission of multicast (i.e., broadcast) messages to members of the system.

Many segments of today's financial and business communities (e.g., stock exchanges, banks, telecommunications companies) require computing environments that are fault tolerant and provide high availability. Downtime in these environments can be extremely costly and are not lightly tolerated. There exists a number of different approaches to providing fault tolerance and high availability. However, one enjoying increasing popularity is the employment of distributed operating systems in connection with a collection of independent processing environments, referred to as nodes, to be connected via some form of a communication interconnect to form a "cluster" which can operate as a single system or as a collection of independent processing resources. High availability and improved fault tolerance are achieved by the distributed nature of the operating system. High availability is achieved by distributing the system services and providing for their failover. With this approach, the system as a whole can still function even with the loss of one or more of the nodes that make up the system.

Regardless of how such processing system clusters are used, it is often advantageous to keep each of the processing elements of such systems up-to-date as, for example, to the system's configuration (e.g., what elements are located where, etc.). This, in turn, often will require that each node possess the capability of transmitting (i.e., broadcasting)

messages to the other nodes of the cluster system. Often, such "multicast" transmissions are sent point-to-point, that is, from a sender node to a first node, then to a second node, and so on until all target nodes have been addressed. This multicast transmission procedure can require considerable processing time, increase the messaging traffic on the communicating medium of the cluster (particularly when the message is intended for every node in the cluster), and impose unacceptable restraints and limitations on system performance. Some procedures will require continuing retransmission of the message when not acknowledged by the intended receiver node. The constant retransmission of the message to all non-responding nodes further increases traffic on the network thereby degrading the overall network performance as well as occupying processor time and other cluster resources.

More importantly, however, is the need to identify the failure to receive a message, i.e., for the intended receiver to determine in some way that a message was sent, but not received. For example, if a sender node and multiple receiver nodes are interconnected by a routing network, it is not unexpected that messages can get lost and not arrive at one or more of the intended receivers (in the case of multicast transmissions). Thus, if the sender node transmits a multicast message that is received by some, but not all, of the intended receivers, those receivers that did not receive the message may well be missing needed information that can inhibit or impede system operation or proper operation of other nodes of the system.

It can be seen, therefore, that there is a need for a more efficient method of multicast transmission in a multiple processor or cluster system that also checks for and supplies possible missing multicast messages.

SUMMARY OF THE INVENTION

The present invention provides a prompt and efficient method of transmitting a multicast message in a multiple processor cluster. In addition, the method provides a technique for sequenced once only delivery of messages supplemented by the ability to supply missing earlier-sent messages to multiple receivers.

The invention finds particular advantage in a multiple processor system is arranged as a cluster of nodes. Each node comprises one or more processor units, although those skilled in this art will readily see that the invention will also find effective use in other multiple processor arrangements. According to the invention, a sender node will initiate a multicast message transmission by inserting, in a destination address field of the message, an address indicative of the message being a multicast transmission. The multicast message is also structured to include a sequence number and a "date of birth" (DOB) marker (a monotonically incrementing value that is indicative of when a node is brought on-line, i.e., comes to life). All nodes communicatively connected to the sender node will, therefore, receive the multicast message, and acknowledge that receipt with a responsive acknowledgement (ACK) message addressed to the sender node (as identified in the multicast message). A failure by the sender node to receive an ACK message from any of the nodes for whom the multicast message was intended within an allotted time period will prompt the sender to assume that the non-responding receiver node(s) did not, for whatever reason, receive the multicast message, and to begin sending point-to-point messages to such nodes. Point-to-point messages will continue to be sent until a responsive ACK is received, or the non-responsive node is declared by the system to be inoperative.

When a multicast message is received, the receiver node will check the sequence number contained in the message. If the sequence number is out of sequence, the receiver node will

queue the message in an ordered queue and then check the queue for the missing least sequence number message, and send a negative acknowledgement for the multicast message corresponding to such missing least sequence number. On the other hand, if the sequence number or the DOB marker contained in the multicast message do not match the sequence number or marker expected by the receiver node, a resynchronization request message will be returned by the receiver node to the sender node. The resynchronization request will cause the sender node to respond with its new marker, and the sequence number of the last multicast message unacknowledged by the receiving node. In this way, lost multicast messages can be accounted for and delivered.

In a further embodiment of the invention, if a receiver node repeatedly fails to respond to multicast messages, but does respond to point-to-point messages, the sender node will make note of that fact, and all future multicast messages will be accompanied by a point-to-point message to that particular receiver node -- until it finally responds to a multicast message.

The invention is disclosed in connection with use of the User Data Protocol (UDP) of the TCP/IP protocol suite to transfer information (messages) between the nodes of the system, including the multicast messages. A multicast message will typically consist of a number of UDP datagrams. Each datagram is provided a sequence number that identifies the location of the datagram in the sequence. Further, subsequent messages as well as prior messages have sequence numbers that identify not only the datagrams with a sequence of a message, but also relative to the other sequences. When a receiving node receives a datagram

A further understanding of the nature and advantages of the inventions herein may be realized by reference to the remaining portions of the specification and the attached drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a simplified representation of a multiple processing system, comprising a number (N) of communicatively interconnected processor nodes, for utilizing the multicast transmission method present invention;

Fig. 2 is a schematic illustration of a multiple processor node of Fig. 1;

Fig. 3A illustrates a multicast message structure showing encapsulation of a message packet used in the present invention in a TCP(UDP)/IP packet which, in turn, may be encapsulated in a local area network (LAN) packet;

Fig. 3B illustrates encapsulation of the message packet used by the present invention in the UDP datagram packet that, in turn, is encapsulated in the message structure shown in Fig. 3A;

Fig. 3C illustrates the fields of the message packet used by the present invention;

Fig. 4 is a flow diagram generally illustrating the steps taken in connection with a multicast message transmission by a (multicast message) sender node of the system shown in Fig. 1;

Fig. 5 is a flow diagram illustrating the steps taken by a receiver node when receiving a multicast message transmission.

DESCRIPTION OF THE PREFERRED EMBODIMENT

Turning now to the figures, and for the moment with specific reference to Fig. 1, there is illustrated a multiple processing system or "cluster" 10 comprising a number of nodes 12 (12_0, 12_1, ..., 12_{n-1}, where n can be as high as 129) - or greater, or less, depending upon the design and implementation

- interconnected by a communication medium 14. Each of the nodes 12 will comprise one, or preferably more (up to 8 for reasons that will be explained below) processor units such as illustrated in Fig. 2 where node 12_0 is shown as comprising four processor units 16 (16_0, 16_1, ..., 16_3), preferably operating under the Windows NT operating system (Windows, NT, and Windows NT are trademarks of MicroSoft Corporation, Redmond, Washington). The NT Windows operating system implements a symmetric multiprocessing (SMP) system for each node 12 having more than one processor unit 16 exploiting the power of multiple processor units through distribution of the NT operating system. It will be evident to those skilled in this art that other operating systems can be employed for an SMP configuration. Further, in connection with the present invention, as will also be evident, the present invention does not require an SMP configuration or a distributed operating system; that is, the method of the present invention can be used in multiple processing systems using one processor unit 16 per node 12. The invention can be advantageously employed in other differently configured multiple processing systems having a need for multicast transmissions.

Although only node 12_0 is shown in Fig. 2 as the representative node, it will be understood that the other nodes 12 of the cluster 10 (Fig. 1) are of the same basic construction as that of node 12_0, with the exception of the number of processor units 16, as indicated above. Thus, a description of node 12_0 should be taken as a description of the other nodes 12 unless otherwise indicated.

As Fig. 2 shows, the processors 16 are communicatively connected to one another and to a shared memory element 20 by a bus architecture 22 of conventional design. In addition, the bus 22 connects to an interface unit 24 that provides the necessary connectivity of the node 12_0 to the communication medium 14. Resident on each of the processors 16 is a monitor (MON) process 26 which, among other things, assists in handling interprocessor communications between that processor

26 and the others. One of those monitor processes 26 is selected as a coordinator monitor (C-MON) process, here C-MON 26a, with the added responsibility of assisting in handling internode communication - particularly multicast transmissions. Each node 12 will have one processor 16 running the C-MON process. The C-MON process 26a of any particular node will also have the responsibility of distributing messages received from the communication medium 14 to the other processors 16 of the node, including message traffic identified as being for a specific processor 16, or received multicast messages, using the NT interprocessor communication service.

Internode communication preferably uses the Internet suite. For message transmissions between one node and another (i.e., a point-to-point transmission), the TCP service may be used due to its reliability. However, point-to-point communication is not particularly suited for multicast message transmissions - particularly in systems with a large number of nodes 12 because of the tendency to absorb the resources of the sender processor 16, and to increase traffic on the communication medium. A more efficient broadcast method is to use connectionless capability of the UDP service which lends itself quite well to multicast transmissions. For multicast transmissions, therefore, the UDP service is preferred because of its adaptability of the connectionless protocol form to multicast transmissions. The inherent unreliability of a connectionless protocol is accounted for by providing a method of determining at the multicast receiving end the ability of determining whether prior multicast message packets were sent but not received by certain of the intended receiver nodes. Thus, any possible degradation in reliability is overcome by the present invention with the incorporations of several safeguards that ensure multicast messages are received by all nodes 16 concerned in a manner that will be described more fully below.

Referring now to Figs. 3A-3C, illustrated is the structure of a multicast message formed for incorporation of the present invention. It is assumed, for purposes of discussing this invention, that the communication medium 14 is structured to operate within a local area network (LAN) of one type or another (e.g., Token-Ring, FDDI, Ethernet, etc.), and that any Internet suite data packet messages will therefore be encapsulated in a LAN packet 30 as Fig. 3A illustrates. Thus, the LAN packet 30 will include an Internet packet 32, adding a LAN header 30a and LAN trailer 30b, and such other information as may be needed by the particular protocol of the LAN. The Internet packet 32, in turn, will comprise a conventional IP header 32a (which will identify the node 12 to which the message is directed), UDP header 326 (in the case of a multicast transmission), and the UDP data area 32c. It is the UDP data area 32c that contains the message packet being transmitted.

The UDP data area 32c is illustrated in greater detail in Fig. 3B, and is shown as including a message header 34a and a data portion 34b. Fig. 3C illustrates in greater detail the fields of the message header 34a, including a type field 38, identifying the message type (e.g., whether a multicast, acknowledgement, negative acknowledgement, etc., message); a destination address field 40 operates to identify the processor 16 of the system 10 (Fig. 1) to which the message packet 36 is directed. If the message packet is a multicast transmission, the IP address and destination address fields are set to a specific value (e.g., minus 1) to identify the message as a multicast transmission.

Continuing with Fig. 3C, the message header 34a further includes a source address field 42 to identify the processor unit 16 sending the message, and a 32-bit sequence number field 44. The MON, C-MON processes 26, 26a of the processor units 16 each maintain a table of sequence numbers, one for each of the other processor units 16 of the system 10. When ever one of the processors 16 sends (point-to-point) a message

packet to another processor 16, the MON process of the sending processor will check the table corresponding to the receiver processor for the next sequence number to use, increment that number by one and insert that sequence number in the message packet. The incremented sequence number is returned to the table to replace the prior (unincremented) sequence number.

In addition, the C-MON process 26a of each node 12 maintains a table that will have one extra entry: a sequence number for multicast transmissions. When a message is being developed for multicast transmission, the table is accessed for the sequence number pertaining to multicast transmissions and, as described above, incremented by one, returned to the table as incremented. The incremented sequence number is placed in the sequence number field 44 of the message packet.

Continuing, the message header 34a is shown as also having a DOB marker field 46. The content of the marker field 46 is indicative of a relative point in time when the sender node 12 came on-line in the system 10, and therefore serves as a kind of "date of birth" for the node. The DOB marker is a monotonically incremented value that is used by each node as a relative birth date. The sequence number and DOB marker (hereinafter, just "marker") of a multicast message are used as a safeguard, as will be seen, against a node 12 failing to receive one or more multicast messages and not knowing that it is missing such message(s).

Finally, the message header 34a includes a process identification (ID) field 48 and a Time Value field 49. The content of process ID field 48 identifies the particular process operating on the sending processor unit 16 (as identified by the content of the destination address field 42) that initiated the multicast transmission. The Time Value field 49 carries what is basically a time stamp that, among other things, allows a sender to develop a run time round-trip interval value. For example, each message transmitted by a sender node 12 will have a time value indicative of the

(local) time of the sender node. The receiver node 12 will typically respond with at least an acknowledgement message that will include, in the Time Value field 49 of the response, the time stamp that was carried by the received message. When the sender node 12 receives back the acknowledgement (or whatever other response), it uses the time stamp value carried in the Time Value field 49 of the acknowledgement or response to determine the run time round-trip interval between the sender and receiver nodes 12 merely subtracting the received time value from the present time value maintained by the sender node 12.

Turning now to Figs 4, and 5, there illustrated, in flow diagram fashion, are the major steps taken to both transmit and receive multicast message transmissions by a sender and receiver nodes 12 (more accurately, the C-MON processes 26a of the sender and receiver nodes 12). Referring first to Fig. 4, the process of sending a multicast message begins, of course, with its preparation in step 62 where such preparation will include a message header 34a to contain: a type value in the Type field 38; a destination address that identifies the message as a multicast message in destination field 40; and, a source address identifying the sender node in the source address field 42. Step 64 determines whether the message transmission is to be multicast, or a transmission to a specific processor unit 16 of the system 10. If the latter, step 66 selects for the Sequence Number field 44, the next sequence number for the intended destination processor unit 16. If a multicast message is being developed, step 68 selects for the sequence address field 44 a sequence number that is greater than the immediately prior multicast message packet sent by this node 12. The DOB marker field 46 will contain, for either transmission type, the DOB marker for this sender node 12, and the process ID field 48 will identify the intended process to receive the transmission, if any. Finally, a time stamp, indicative of the local time of transmission will be put in the Time Value field 49.

Step 70 sends the message encapsulated within an Internet packet format as the data payload, and in turn encapsulated in whatever additional packaging is needed by the protocol used for communication using the communication medium 14 (Fig. 1). Thus, the Internet packet may be encapsulated in a LAN packet - as illustrated in Fig. 3A. The sender node 12 (i.e., the C-MON process 26a of that node) will then set a timer and move to step 74 where it will wait for receipt of acknowledgements (ACKs) (not shown), from each intended receiver node 12 confirming receipt of the multicast transmission, or non-acknowledgements (ACK) that, in effect, request a point-to-point re-transmission of the message.

The sender C-MON process 26a will maintain a log of the receiver nodes intended to receive the multicast message. As the sender C-MON process 26a is notified of the proper receipt the multicast transmission by an ACK, it will remove the acknowledging sender node identification from the log.

Digressing for the moment, ACK and NAK messages are of the same basic form shown in Figs. 3A-3C. The destination address field 40 will identify the multicast sender node 12, the source address field 42 will identify the multicast receiver node sending the ACK or NAK, and the sequence number field 44 will carry the sequence number of the multicast message that the ACK or NAK message is responding, and the marker field 46 carries the marker of the receiver node 12 sending the ACK or NAK message.

If an ACK message is received (step 74) from one of the intended receiver nodes 12, the multicast send procedure will move to step 77, where the identification of the receiving node sending the ACK (as contained in the source field 42 of the ACK message) is deleted from a list of nodes from which the multicast sender node is awaiting acknowledgement, and the procedure returns to await any remaining ACKs. On the other hand, if a NAK message is received (step 76), the sender node will, determine from the received NAK message which multicast

message is requested by the receiver node sending the NAK, and send it as a point-to-point message transmission. The procedure then returns to await the acknowledgement of the just sent message and any others still outstanding from earlier sent multicast messages.

The timer (not shown) that is set in the multicast message send step 70 will eventually time out, if all the intended receiving nodes of a multicast transmission have not acknowledged the transmission within the determined run time computed round-trip interval. Thereafter, the multicast sender node 12 will begin resending the multicast message point-to-point, to those nodes not yet providing ACK or NAK messages (steps 82-84), and continue doing so until the sender node 12 receives an ACK message from all the nodes or the non-respond node(s) 12 is declared to have been removed from the system (step 82). When all nodes have responded (or those that haven't are declared to no longer be in the system) the multicast transmission will considered to be concluded (step 86).

Fig. 5 broadly outlines the procedure that takes place at a receiver node 12 when receiving a multicast message, beginning with the receipt of the multicast message in step 102 by the C-MON 26a of the receiving node. Each receiver node 12 will maintain a queue, one queue for each node of the system 10, of all received multicast messages from each of the other nodes 12. when a multicast message is received, it is placed on the queue for the sender node (as identified by the Source Address field 42 of the message). Later, the message will be examined, in step 104, to determine first if the DOB marker of sender node included in the message (Marker field 46) is different from that obtained from previous multicast messages from this particular sender node. (If this is the first marker from this sender, the receiving node, i.e., the C-MON process 26a, will store the marker for this sender.) If there is a difference, the receiver node will, in step 108, prepare a resynchronization message that is sent to the multicast sender node, requesting a resynchronization.

A resynchronization request by a receiver node will be responded to by the multicast sender node with the sequence number of the earliest unacknowledged multicast message sent by that sender node. The receiver node, when receiving the resynchronization response, will check its queue for multicast messages from that sender node that are missing, as indicated by missing sequence numbers between that contained in the resynchronization response message and the most recently received multicast message. The sender node will then send a negative acknowledgement (NAK) for each missing multicast message (if a NAK message has not been sent within a run time round-trip interval earlier).

If the DOB marker contained in the received multicast message matches that expected by the receiver node, the receiver node 12 will, at step 110, check the sequence number to ensure that the sequence number is as expected, i.e., that it is next in the sequence of multicast messages received from this sender node 12. If so, the receiver will send an ACK message (point-to-point) to the multicast sender node (step 112), and then conclude in step 114.

If the sequence number does not match, the receiver node 12 will check (step 116) to see if the sequence number is the same as an earlier received multicast message, i.e., is this received multicast message a duplicate of one earlier received. If so, the receiving node 12 will disregard the message, and moving to step 112 to send an ACK for the message, and concluding at step 114.

If the receiver node determines, in steps 110 and 116, that the sequence number is not as expected, and not a duplicate of an earlier received multicast message from this sender, the receiving node will then check its queue of received multicast messages for the multicast sender node to find the missing least sequence number multicast message (i.e., the earliest missing multicast message form this sender node 12). If a NAK has not been sent for this missing least

sequence number multicast message within a run time computed round-trip interval, one will be sent, and the receiving node will move to the conclusion step to await response in the form of the requested message.

As indicated above, multicast messages received by a receiver node are placed in an ordered queue. The C-MON process 26a will check the queue, and all available messages up to the last or missing multicast message will be delivered to the other MON process 26 of that node via the NT interprocessor communication service.

In summary, there has been disclosed a multicast method for a TCP/IP based network in which all multicast packets are sent as UDP datagrams using a preconfigured broadcast IP address. These datagrams are received on a broadcast IP port by the Coordinator monitor (C-MON) process who will relay these datagrams to other monitor (MON) processes executing in different processor units of the same node. An acknowledgment is sent using a UDP datagram to the sender. If an acknowledgment is not received within a run-time computed retry interval, the sender will keep retrying a point-to-point UDP datagram until the remote end (receiver) is declared dead by the cluster manager. Once all alive remote ends acknowledge the packet, the sender is notified for successful delivery of the multicast.

A multicast coordinator is a MON process whose UDP port is designated as the broadcast port. This selection is made through the Windows NT registry. If no MON is designated as a multicast coordinator then the first MON process with respect to NT registry configuration is designated as multicast coordinator.

A multicast packet is sent on a broadcast port of pre-configured broadcast IP address. Since a TCP/IP based network may have been configured to limit the broadcast within a subnet, the broadcast on TCP/IP network may not get delivered

to all the nodes participating in a cluster. The sender waits for the acknowledgment from all the nodes of the cluster. If an acknowledgment is not received within a run-time computed retry interval, a point-to-point UDP datagram is sent to non-responding Coordinator MON. The sender keeps on retrying using exponentially incrementing retry interval until an acknowledgment is received from the target node or the target is declared dead. Since the target IP port for the target node may be configured to receive non-multicast packets, the header of a multicast packet is set to indicate a multicast packet.

Once a multicast packet is received by a coordinator MON, it is checked for duplicate or out of sequence delivery. If this is a duplicate packet a positive acknowledgment is sent to the sender and no other action is taken. On the other hand if an out of sequence packet is received, the packet is queued in an ordered queue and then the queue is checked for the missing least sequence number packet. If a negative acknowledgment for the missing sequence number has not been sent within a run time computed round-trip interval then a negative acknowledgment for the required sequence number packet is sent.

Once a packet is queued in the ordered queue, the queue is checked by the receiver node Coordinator MON and all available packets up to the last expected/missing packets are delivered to other MON using Windows NT operating system's interprocessor communication mechanism.

If a positive acknowledgment is received by the sender then all packets waiting for the acknowledgment up to the sequence number as indicated by the positive acknowledgment are assumed to be acknowledged from the remote end. The sender checks the list of nodes that should have acknowledged this multicast. If all the nodes have acknowledged this packet then multicast is declared completed.

Each multicast packet carries a sequence number and a monotonically incrementing marker. If either sequence number or the marker do not match with the expected sequence number or the expected marker, a re-synchronize request packet is generated by the receiver and sent to the multicast sender. Upon reception of the re-synchronize request packet, a re-synchronize response is sent with the new marker and last un-acknowledged multicast packet sequence number from the receiver. A re-synchronization response can be initiated when a negative acknowledgment received for a non-waiting multicast packet.

WHAT IS CLAIMED IS:

1. A method for multicast transmission of a message to a plurality of processing nodes, the method including the steps of:

a sender processing node transmitting a multicast message for receipt by the plurality of processing nodes;

re-sending the multicast message individually to each of the processing nodes not providing the acknowledgment after a predetermined time interval.

2. The method of claim 1, wherein the predetermined time interval is a calculated specific run-time interval determined separately for each of the plurality of processor nodes.

3. The method of claim 1 wherein the re-sending step includes re-sending the multicast message to each of the processing nodes until an acknowledgement is received.

4. The method of claim 3, wherein re-sending of the multicast message is in exponentially incrementing time intervals.

5. The method of claim 1, wherein the sender processing node transmits a number of multicast messages each including a sequence number indicative position of such multicast message within the number of multicast messages; and further including the steps:

each of the processing nodes receiving each of the number of multicast message checking the sequence number of each multicast message to ensure all prior multicast messages have been received by such processing node.

6. The method of claim 1, the sender processor node including in the multicast message a marker indicative of a life of operation of the sender processor node, each of the plurality of processor nodes receiving the multicast message checking the marker against a prior received marker.

7. The method of claim 6, wherein each of the plurality of processing nodes transmits to the sender processor node a resynchronization message if the marker and the prior received marker do not match.

8. In a processing system of a type comprising a plurality of processor nodes communicatively interconnected for communication therebetween, a method of transmitting sequence of multicast messages from a sender one of the processor nodes to other of the processor nodes that includes the steps of:

including in each of the sequence of multicast messages a sequence number indicative of the position of such multicast message with the sequence;

each of the other processor nodes receiving the multicast messages checking the sequence number against a prior received one of the sequence of multicast messages, and sending to the sender processor a no-acknowledgement message identifying a missing multicast message;

upon receipt of the no-acknowledgement message, the sender processor re-sending the missing multicast message.

9. The method of claim 8, wherein included in each of the sequence of multicast messages a marker indicative of a time of life of the sender processor unit, and wherein each of the other processor nodes receiving the sequence of multicast messages compares the marker to a prior-received marker, and sending a resynchronization message to the sender processor node if the mark and the prior-received marker to not compare, the sender processor node resending all prior unacknowledged ones of the sequence of multicast messages.

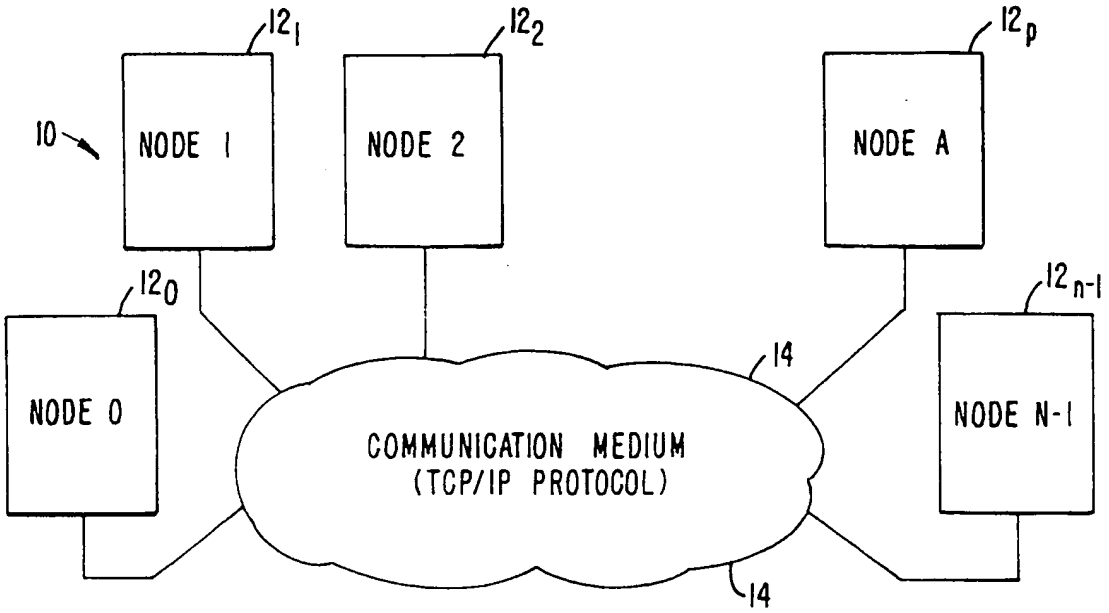
10
NODE 1
NODE 2
NODE A
NODE 0
NODE N-1
COMMUNICATION MEDIUM
(TCP/IP PROTOCOL)
14
14
12_1
12_2
12_p
12_0
12_{n-1}

FIG. 1.

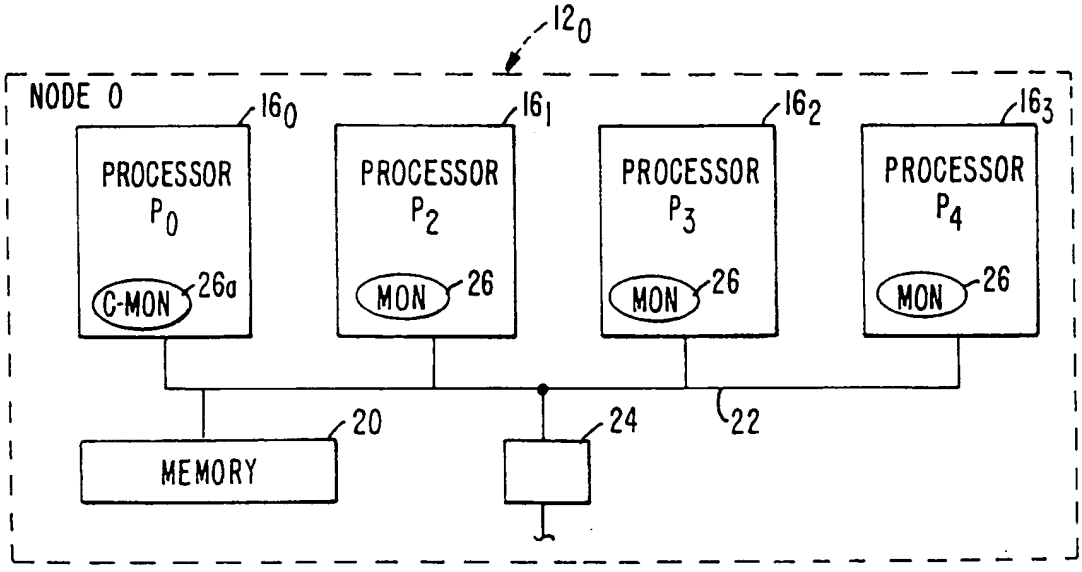
12_0
NODE 0
16_0
16_1
16_2
16_3
PROCESSOR P_0
PROCESSOR P_2
PROCESSOR P_3
PROCESSOR P_4
C-MON
26a
MON
26
MON
26
MON
26
MEMORY
20
24
22

FIG. 2.

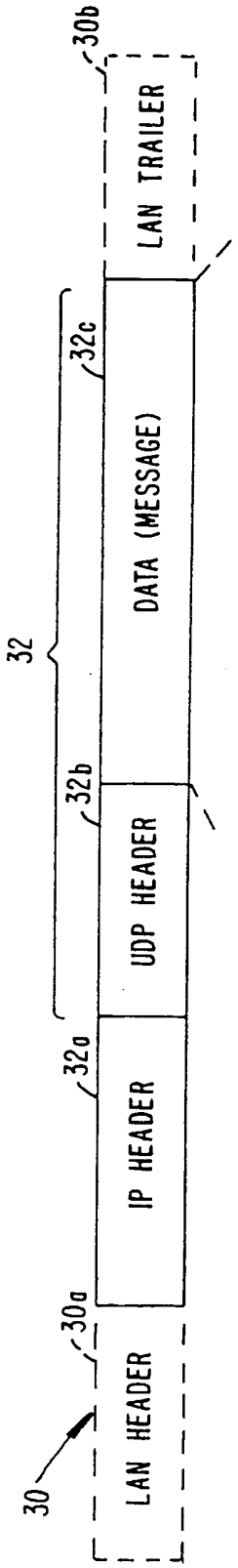

FIG. 3A.

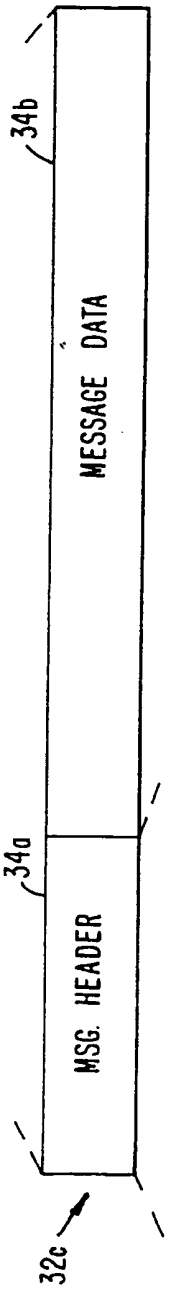

FIG. 3B.

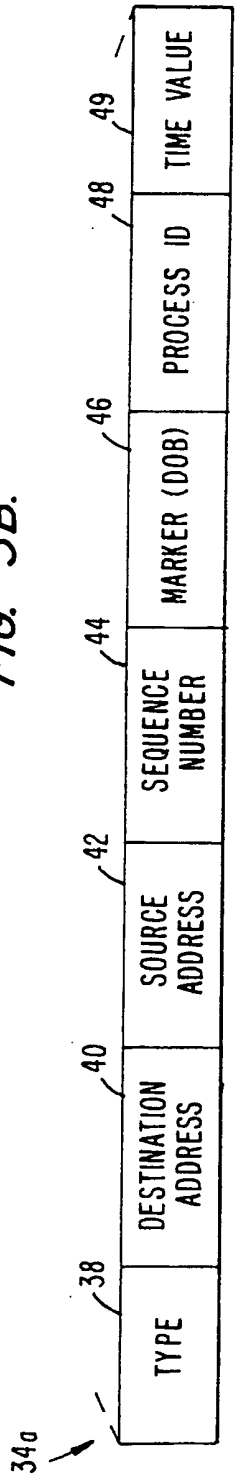

FIG. 3C.

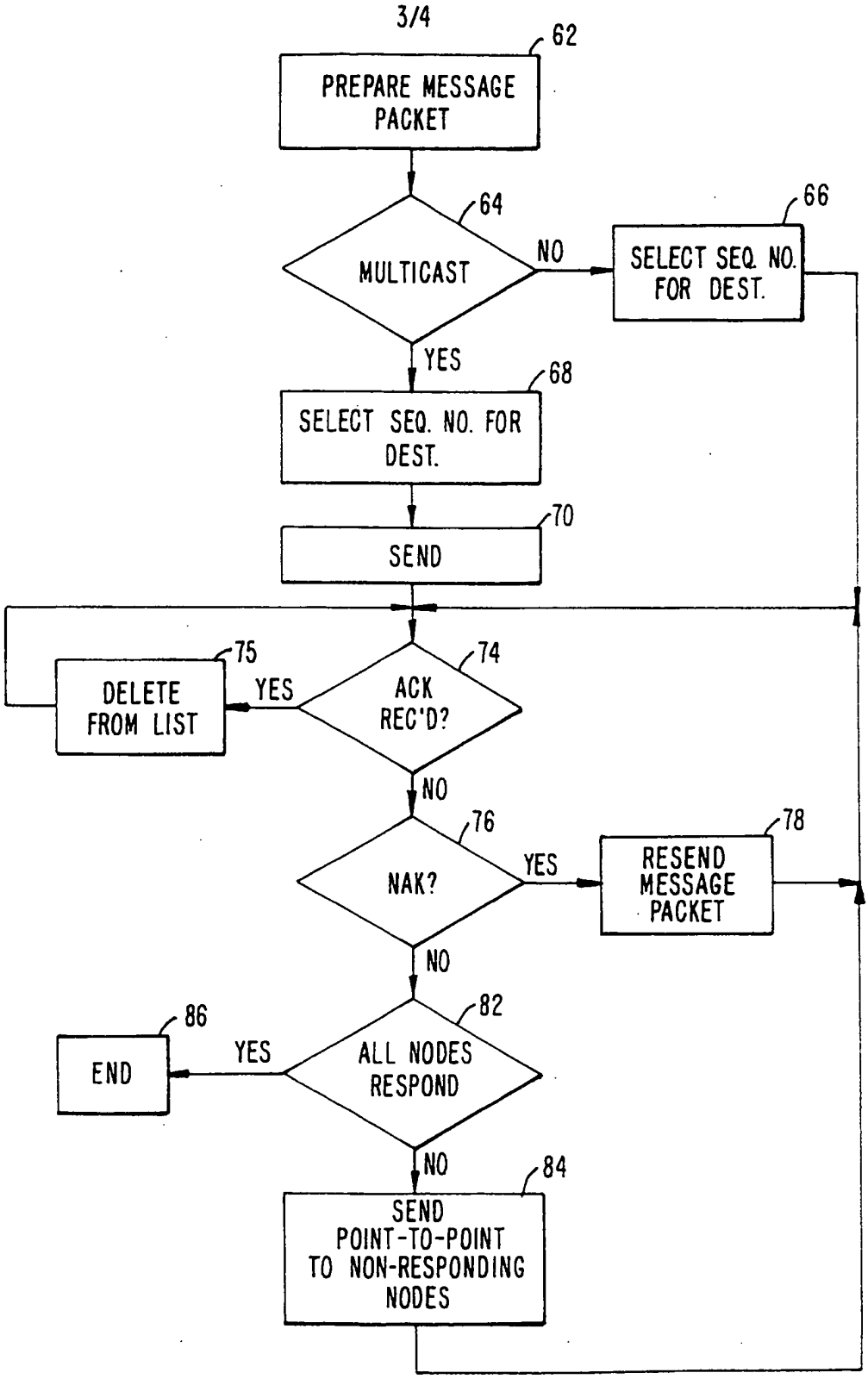
62
PREPARE MESSAGE PACKET
64
MULTICAST
NO
66
SELECT SEQ. NO. FOR DEST.
YES
68
SELECT SEQ. NO. FOR DEST.
70
SEND
75
DELETE FROM LIST
YES
74
ACK REC'D?
NO
76
NAK?
YES
78
RESEND MESSAGE PACKET
NO
82
ALL NODES RESPOND
86
END
YES
NO
84
SEND POINT-TO-POINT TO NON-RESPONDING NODES

FIG. 4.

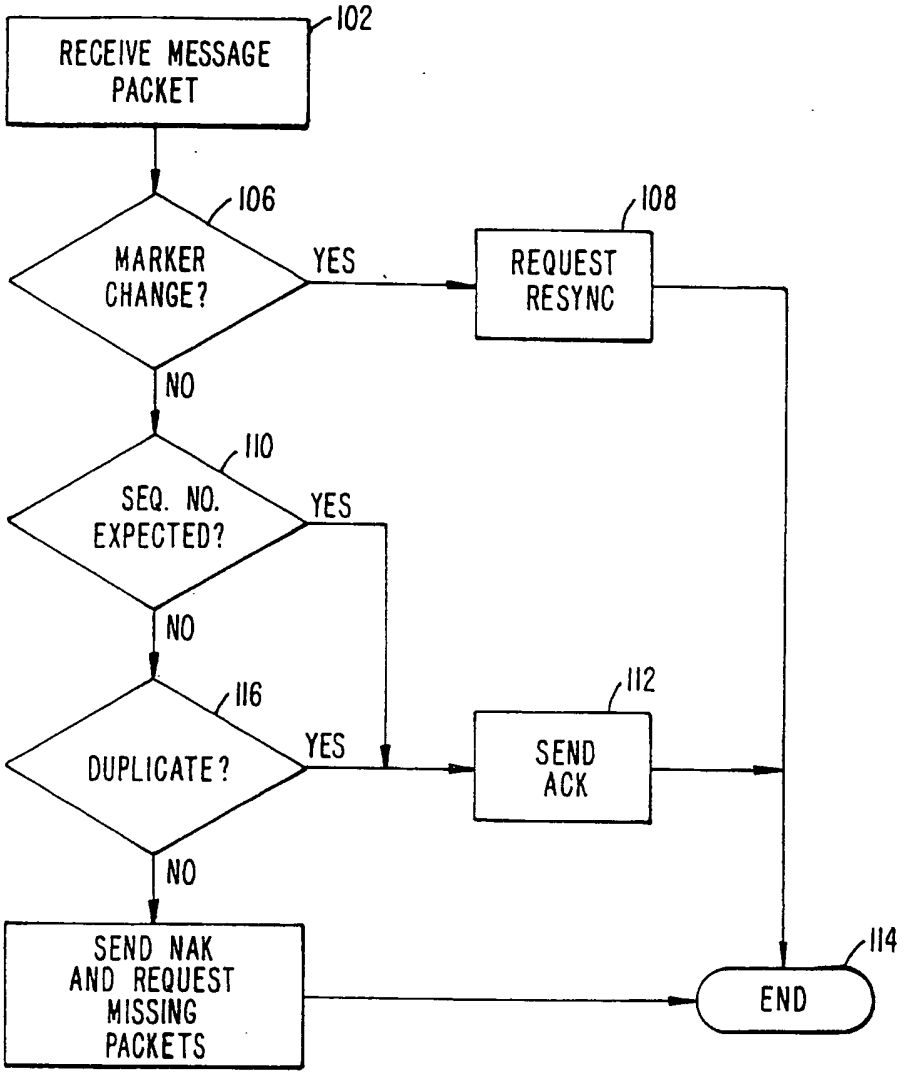
RECEIVE MESSAGE PACKET
102
MARKER CHANGE?
106
YES
REQUEST RESYNC
108
NO
SEQ. NO. EXPECTED?
110
YES
NO
DUPLICATE?
116
YES
SEND ACK
112
NO
SEND NAK AND REQUEST MISSING PACKETS
END
114

FIG. 5.